Abstract
In entity linking, mentions of named entities in raw text are disambiguated against a knowledge base (KB). This work focuses on linking to unseen KBs that do not have training data and whose schema is unknown during training. Our approach relies on methods to flexibly convert entities with several attribute-value pairs from arbitrary KBs into flat strings, which we use in conjunction with state-of-the-art models for zero-shot linking. We further improve the generalization of our model using two regularization schemes based on shuffling of entity attributes and handling of unseen attributes. Experiments on English datasets where models are trained on the CoNLL dataset, and tested on the TAC-KBP 2010 dataset show that our models are 12% (absolute) more accurate than baseline models that simply flatten entities from the target KB. Unlike prior work, our approach also allows for seamlessly combining multiple training datasets. We test this ability by adding both a completely different dataset (Wikia), as well as increasing amount of training data from the TAC-KBP 2010 training set. Our models are more accurate across the board compared to baselines.

1 Introduction
Entity linking consists of linking mentions of entities found in text against canonical entities found in a target knowledge base (KB). Early work in this area was motivated by the availability of large KBs with millions of entities (Bunescu and Pasca, 2006). Most subsequent work has followed this tradition of linking to a handful of large, publicly available KBs such as Wikipedia, DBPedia (Auer et al., 2007) or the KBs used in the now decade-old TAC-KBP challenges (McNamee and Dang, 2009; Ji et al., 2010). As a result, previous work always assumes complete knowledge of the schema of the target KB that entity linking models are trained for, i.e. how many and which attributes are used to represent entities in the KB. This allows training supervised machine learning models that exploit the schema along with labeled data that link mentions to this a priori known KB. However, this strong assumption breaks down in scenarios which require linking to KBs that are not known at training time. For example, a company might want to automatically link mentions of its products to an internal KB of products that has a rich schema with several attributes such as product category, description, dimensions, etc. It is very unlikely that the company will have training data of this nature, i.e. mentions of products linked to its database.

Our focus is on linking entities to unseen KBs with arbitrary schemas. One solution is to annotate data that can be used to train specialized models for each target KB of interest, but this is not scalable. A more generic solution is to build entity linking models that work with arbitrary KBs. We follow this latter approach and build entity linking models that link to target KBs that have not been observed during training.¹ Our solution builds on recent models for zero-shot entity linking (Wu et al., 2020; Logeswaran et al., 2019). However, these models assume the same, simple KB schema during training and inference. We generalize these models to handle different KBs during training and inference, containing entities represented with an arbitrary set of attribute-value pairs.

¹“Unseen KBs” refers to scenarios where we neither know the entities in the KB, nor its schema.
Table 1: This table compares the entity linking framework in the present work with those in previous work.

| Test entities not seen during training | ✓ | ✓ | ✓ |
| Test KB schema unknown | ✓ | ✓ |
| Out-of-domain test data | ✓ | ✓ |
| Unrestricted Candidate Set | ✓ | ✓ | ✓ |

Table 1 summarizes differences between our framework and those from prior work.

Contextualized Representations for Entity Linking Models in this work are based on BERT (Devlin et al., 2019). While many studies have tried to explain the effectiveness of BERT for NLP tasks (Rogers et al., 2020), the work by Tenney et al. (2019) is most relevant as they use probing tasks to show that BERT encodes knowledge of entities. This has also been shown empirically by many works that use BERT and other contextualized models for entity linking and disambiguation (Broscheit, 2019; Shahbazi et al., 2019; Yamada et al., 2020; Févry et al., 2020; Poerner et al., 2020).
The attributes \( k_i \) collectively form the **schema** of \( \mathcal{KB} \). The disambiguation of each \( m \in \mathcal{M} \) is aided by the context \( c \) in which \( m \) appears.

Models for entity linking typically consist of two stages that balance recall and precision.

1. **Candidate generation**: The objective of this stage is to select \( K \) candidate entities \( \mathcal{E} \subset \mathcal{KB} \) for each mention \( m \in \mathcal{M} \), where \( K \) is a hyperparameter and \( K << |\mathcal{KB}| \). Typically, models for candidate generation are less complex (and hence, less precise) than those used in the following (re-ranking) stage since they handle all entities in \( \mathcal{KB} \). Instead, the goal of these models is to produce a small but high-recall candidate list \( \mathcal{E} \). Ergo, the success of this stage is measured using a metric such as recall@\( K \) i.e. whether the candidate list contains the correct entity.

2. **Candidate Re-ranking**: This stage ranks the candidates in \( \mathcal{E} \) by how likely they are to be the correct entity. Unlike candidate generation, models for re-ranking are typically more complex and oriented towards generating a high-precision ranked list since the objective of this stage is to identify the most likely entity for each mention. This stage is evaluated using precision@1 (or accuracy) i.e. whether the highest ranked entity is the correct entity.

In traditional entity linking, the training mentions \( \mathcal{M}_{\text{train}} \) and test mentions \( \mathcal{M}_{\text{test}} \) both link to the same KB. Even in the zero-shot settings of Logeswaran et al. (2019), while the training and target domains and KBs are mutually exclusive, the schema of the KB is constant and known. On the contrary, our goal is to link test mentions \( \mathcal{M}_{\text{test}} \) to a knowledge base \( \mathcal{KB}_{\text{test}} \) which is not known during training. The objective is to train models on mentions \( \mathcal{M}_{\text{train}} \) that link to \( \mathcal{KB}_{\text{train}} \) and directly use these models to link \( \mathcal{M}_{\text{test}} \) to \( \mathcal{KB}_{\text{test}} \).

### 3.2 Zero-shot Entity Linking

The starting point (and baselines) for our work are the state-of-the-art models for zero-shot entity linking, which we briefly describe here (Wu et al., 2020; Logeswaran et al., 2019).

**Candidate Generation** Our baseline candidate generation approach relies on similarities between mentions and candidates in a vector space to identify the candidates for each mention (Wu et al., 2020) using two BERT models. The first BERT model encodes a mention \( m \) along with its context \( c \) into a vector representation \( \mathbf{v}_m \). \( \mathbf{v}_m \) is obtained from the pooled representation captured by the [CLS] token used in BERT models to indicate the start of a sequence. In this encoder, a binary (0/1) indicator vector is used to identify the mention span. The embeddings for this indicator vector (indicator embeddings) are added to the token embeddings of the mention as in Logeswaran et al. (2019). The second unmodified BERT model (i.e. not containing the indicator embeddings as in the mention encoder) independently encodes each \( e \in \mathcal{KB} \) into vectors. The candidates \( \mathcal{E} \) for a mention are the \( K \) entities whose representations are most similar to \( \mathbf{v}_m \). Both BERT models are fine-tuned jointly using a cross-entropy loss to maximize the similarity between a mention and its corresponding correct entity, when compared to other random entities.

**Candidate Re-ranking** The candidate re-ranking approach uses a BERT-based cross-attention encoder to jointly encode a mention and its context along with each candidate from \( \mathcal{E} \) (Logeswaran et al., 2019). Specifically, the mention \( m \) is concatenated with its context on the left (\( c_l \)), its context on the right (\( c_r \)), and a single candidate entity \( e \in \mathcal{E} \). An [SEP] token, which is used in BERT to separate inputs from different segments, is used here to separate the mention in context, from the candidate. This concatenated string is encoded using BERT\(^3\) to obtain, \( \mathbf{h}_{m,e} \) a representation for this mention/candidate pair (from the [CLS] token). Given a candidate list \( \mathcal{E} \) of size \( K \) generated in the previous stage, \( K \) scores are generated for each mention, which are subsequently scored using a dot-product with a learned weight vector (\( w \)). Thus,

\[
\text{score}_{m,e} = \mathbf{w}^T \mathbf{h}_{m,e}.
\]

The candidate with the highest score is chosen as the correct entity, i.e.

\[
e^* = \arg \max_{k=1}^{K} \text{score}_{m,e_k}.
\]

\(^2\)We re-implemented these models and verified them by comparing results with those in the original papers.

\(^3\)This BERT model also contains the indicator embeddings as described in candidate generation.
4 Linking to Unseen Knowledge Bases

The models in Section 3 were designed to operate in settings where the entities in the target KB were only represented using a textual description. For example, the entity Douglas Adams would be represented in such a database using a description as follows: “Douglas Adams was an English author, screenwriter, essayist, humorist, satirist and dramatist. He was the author of The Hitchhiker’s Guide to the Galaxy.”

However, linking to unseen KBs requires handling entities with an arbitrary number and type of attributes. The same entity (Douglas Adams) can be represented in a different KB using attributes such as “name”, “place of birth”, etc. (top of Figure 1). This raises the question of whether such models, that harness the power of pre-trained language models, generalize to linking mentions to unseen KBs, including those without such textual descriptions. This section presents multiple ideas to this end.

4.1 Representing Arbitrary Entities using Attribute Separators

One way of using these models for linking against arbitrary KBs is by defining an attribute-to-text function $f$, that maps arbitrary entities with any set of attributes $\{k_i, v_i\}_{i=1}^n$ to a string representation $e$ that can be consumed by BERT, i.e.

$$e = f(\{k_i, v_i\}_{i=1}^n).$$

If all entities in the KB are represented using such string representations, then the models described in Section 3 can directly be used for arbitrary schemas. This leads to the question: how can we generate string representations for entities from arbitrary KBs such that they can be used for BERT-based models? Alternatively, what form can $f$ take?

A simple answer to this question is concatenation of the values $v_i$, given by

$$f(\{k_i, v_i\}_{i=1}^n) = v_1 v_2 \ldots v_n.$$

We can improve on this by adding some structure to this representation by teaching our model that the $v_i$ belong to different segments. As in the baseline candidate re-ranking model, we do this by separating them with [SEP] tokens. We call this [SEP]-separation. This approach is also used by Logeswaran et al. (2019) and Mulang’ et al. (2020) who observed during training the entity linking model(s) introduces special tokens based on frequently occurring attributes (which in this example are “name” and “occupation”).

$$f(\{k_i, v_i\}_{i=1}^n) = [SEP] v_1 [SEP] v_2 \ldots [SEP] v_n.$$

The above two definitions of $f$ use the values $v_i$, but not the attributes $k_i$, which also contain meaningful information. For example, if an entity seen during inference has a capital attribute with the value “New Delhi”, seeing the capital attribute allows us to infer that the target entity is likely to be a place, rather than a person, especially if we have seen the capital attribute during training. We capture this information using attribute separators, which are reserved tokens (in the vein of [SEP] tokens) corresponding to attributes. In this case, $f$ takes the form

$$f(\{k_i, v_i\}_{i=1}^n) = [K_1] v_1 [K_2] v_2 \ldots [K_n] v_n.$$

These $[K_i]$ tokens are not part of the default BERT vocabulary. Hence, we augment the default vocabulary with these new tokens and introduce them during training the entity linking model(s) based on the most frequent attribute values seen in the target KB of the training data, and randomly initialize their token embeddings. During inference, when faced with an unseen KB, we use attribute separators for only those attributes that have been observed during training, and use the [SEP] token for the remaining attributes.

Figure 1 illustrates the three instantiations of $f$. In all cases, attribute-value pairs are ordered in descending order of the frequency with which they appear in the training KB. Finally, since both
the candidate generation and candidate re-ranking models we build on use BERT, the techniques discussed here can be applied to both stages, but we only focus on re-ranking.

4.2 Regularization Schemes for Improving Generalization

Building models for entity linking against unseen KBs requires that such models do not overfit to the training data by memorizing characteristics of the training KB. This is done by using two regularization schemes that we apply on top of the candidate string generation techniques discussed in the previous section.

The first scheme, which we call attribute-OOV, prevents models from overtly relying on individual $[K_i]$ tokens and generalize to attributes that are not seen during training. Analogous to how out-of-vocabulary tokens are commonly handled (Dyer et al., 2015, *inter alia*), every $[K_i]$ token is stochastically replaced with the [SEP] token during training with probability $p_{drop}$. This encourages the model to encode semantics of the attributes in not only the $[K_i]$ tokens, but also in the [SEP] token, which is used when unseen attributes are encountered during inference.

The second regularization scheme discourages the model from memorizing the order in which particular attributes occur. Under attribute-shuffle, every time an entity is encountered during training, its attribute/values are randomly shuffled before it is converted to a string representation using the techniques from Section 4.1.

5 Experiments and Discussion

5.1 Data

Our held-out test bed is the TAC-KBP 2010 data (LDC2018T16) which consists of documents from English newswire, discussion forum and web data (Ji et al., 2010). The target KB ($KB_{test}$) is the TAC-KBP Reference KB and is built from English Wikipedia articles and their associated infoboxes (LDC2014T16). Our primary training and validation data is the CoNLL-YAGO dataset (Hoffart et al., 2011), which consists of documents from the CoNLL 2003 Named Entity Recognition task (Tjong Kim Sang and De Meulder, 2003) linked to multiple KBs. To ensure that our training and target KBs are different, we use Wikidata as our training KB. Specifically, we use the subset of entities from Wikidata with a Wikipedia page. We ignore all mentions without a corresponding entity in the KB, both during training and inference, leaving the task of handling such NIL entities to future work. Finally, we use the Wikia dataset (Logeswaran et al., 2019) for experiments that investigate the impact of multiple datasets (Section 5.5).

Table 2 describes the sizes of these various datasets along with the number of entities in their respective KBs.

<table>
<thead>
<tr>
<th></th>
<th>Number of mentions</th>
<th>Size of target KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoNLL-YAGO (train)</td>
<td>18.5K</td>
<td>5.7M</td>
</tr>
<tr>
<td>CoNLL-YAGO (val.)</td>
<td>4.8K</td>
<td></td>
</tr>
<tr>
<td>Wikia (train)</td>
<td>49.3K</td>
<td>0.5M</td>
</tr>
<tr>
<td>Wikia (val.)</td>
<td>10.0K</td>
<td></td>
</tr>
<tr>
<td>TAC KBP 2010 (test)</td>
<td>1.7K</td>
<td>0.8M</td>
</tr>
</tbody>
</table>

Table 2: Number of mentions in our training, validation, and test sets, along with the number of entities in their respective KBs.

While covering similar domains, Wikidata and the TAC-KBP Reference KB have different schemas. Wikidata is more structured and entities are associated with statements represented using attribute-value pairs, which are short snippets rather than full sentences. The TAC-KBP Reference KB contains both short snippets like these, along with the text of the Wikipedia article of the entity. The two KBs also differ in size, with Wikidata containing almost seven times the number of entities in TAC KBP.

Both during training and inference, we only retain the 100 most frequent attributes in the respective KBs. The attribute-separators (Section 4.1) are created corresponding to the 100 most frequent attributes in the training KB. Candidates and mentions (with context) are represented using strings of 128 sub-word tokens each, across all models.

---


8. [https://github.com/lajanugen/zeshel](https://github.com/lajanugen/zeshel)
5.2 Hyperparameters

All BERT models are uncased BERT-base models with 12 layers, 768 hidden units, and 12 heads with default parameters, and trained on English Wikipedia and the BookCorpus. The probability \( p_{\text{drop}} \) for attribute-OOV is set to 0.3. Both candidate generation and re-ranking models are trained using the BERT Adam optimizer (Kingma and Ba, 2015), with a linear warmup for 10% of the first epoch to a peak learning rate of \( 2 \times 10^{-5} \) and a linear decay from there till the learning rate approaches zero.\(^9\) Candidate generation models are trained for 200 epochs with a batch size of 256. Re-ranking models are trained for 4 epochs with a batch size of 2, and operate on the top 32 candidates returned by the generation model. Hyperparameters are chosen such that models can be run on a single NVIDIA V100 Tensor Core GPU with 32 GB RAM, and are not extensively tuned. All models have the same number of parameters except the ones with attribute-separators which have 100 extra token embeddings (of size 768 each).

**Candidate generation** Since the focus of our experiments is on re-ranking, we use a fixed candidate generation model for all experiments that combines the architecture of Wu et al. (2020) (Section 3) with [SEP]-separation to generate candidate strings. This model also has no knowledge of the test KB and is trained only once on the CoNLL-Wikidata dataset. It achieves a recall@32 of 91.25 when evaluated on the unseen TAC-KBP 2010 data.

5.3 Research Questions

We evaluate the re-ranking model (Section 3) in several settings to answer the following questions:

1. Do the attribute-to-text functions (Section 4.1) generate useful string representations for arbitrary entities? Specifically, can these representations be used with the re-ranking model (Section 3) to link to the unseen \( KB_{\text{test}} \)?
2. Do all three key components—attribute-separators (Section 4.1), attribute-shuffling, and attribute-OOV (Section 4.2)—contribute equally to the final model?
3. Does training on more than one KB with different schemas help models in more accurately linking to \( KB_{\text{test}} \)?

4. Do improvements for generalizing to unseen \( KB_{\text{test}} \) also translate to scenarios where there is training data that also links to \( KB_{\text{test}} \)?

For all experiments, we report the mean and standard deviation of the accuracy across five runs with different random seeds.

5.4 Main results

Our primary experiments focus on the first two research questions and study the accuracy of the model that uses the re-ranking architecture from Section 3 with the three core components introduced in Section 4 viz. attribute-separators to generate string representations of candidates, along with attribute-OOV and attribute-shuffle for regularization. We compare this against two baselines without these components that use the same architecture and use concatenation and [SEP]-separation instead of attribute-separators. As a reminder, all models are trained as well as validated on CoNLL-Wikidata and evaluated on the completely unseen TAC-KBP 2010 data.

Results confirm that adding structure to the candidate string representations via [SEP] tokens leads to more accurate models compared to generating strings by concatenation (Table 3). Using attribute-separators instead of [SEP] tokens leads to an absolute gain of over 5% and handling unseen attributes via attribute-OOV further increases the accuracy to 56.2%, a 7.1% increase over the [SEP] baseline. These results show that the attribute-separators capture meaningful information about attributes, even when only a small number of attributes from the training data (15) are observed during inference.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>concatenation</td>
<td>47.2 ± 7.9</td>
</tr>
<tr>
<td>[SEP]-separation</td>
<td>49.1 ± 2.6</td>
</tr>
<tr>
<td>attribute-separation (no reg.)</td>
<td>54.7 ± 3.8</td>
</tr>
<tr>
<td>++attribute-OOV</td>
<td>56.2 ± 2.5</td>
</tr>
<tr>
<td>++attribute-shuffle</td>
<td>58.2 ± 3.6</td>
</tr>
<tr>
<td>++attribute-OOV + shuffle</td>
<td>61.6 ± 3.6</td>
</tr>
</tbody>
</table>

Table 3: Training on CoNLL-Wikidata and testing on the TAC-KBP 2010 test set reveals that using attribute-separators instead of [SEP] tokens yields string representations for candidates that result in more accurate models. Regularization schemes (Section 4.2) further improve accuracy to 61.6% without using any training data from the test KB.
Shuffling attribute-value pairs before converting them to a string representation using attribute-separators also independently provides an absolute gain of 3.5% over the model which uses attribute-separators without shuffling. Overall, models that combine attribute-shuffling and attribute-OOV are the most accurate with an accuracy of 61.6%, which represents a 12% absolute gain over the best baseline model.

Prior work (Raiman and Raiman, 2018; Cao et al., 2018; Wu et al., 2020; Févry et al., 2020) reports higher accuracies on the TAC data but they are fundamentally incomparable with our numbers due to the simple fact that we are solving a different task with three key differences: (1) Models in prior work are trained and evaluated using mentions that link to the same KB. On the contrary, we show how far we can go without such in-KB training mentions. (2) The test KB used by these works is different from our test KB. Each entry in the KB used by prior work simply consists of the name of the entity with a textual description, while each entity in our KB is represented via multiple attribute-value pairs. (3) These models exploit the homogeneous nature of the KBs and usually pre-train models on millions of mentions from Wikipedia. This is beneficial when the training and test KBs are Wikipedia or similar, but is beyond the scope of this work, as we build models applicable to arbitrary databases.

### 5.5 Training on multiple unrelated datasets

An additional benefit of being able to link to multiple KBs is the ability to train on more than one dataset, each of which links to a different KB with different schemas. While prior work has been unable to do so due to its reliance on knowledge of \( KB_{test} \), this ability is more crucial in the settings we investigate, as it allows us to stack independent datasets for training. This allows us to answer our third research question. Specifically, we compare the [SEP]-separation baseline with our full model that uses attribute-separators, attribute-shuffle, and attribute-OOV. We ask whether the differences observed in Table 3 also hold when these models are trained on a combination of two datasets viz. the CoNLL-Wikidata and the Wikia datasets, before being tested on the TAC-KBP 2010 test set.

Adding the Wikia dataset to training increases the accuracy of the full model by 6%, from 61.6% to 66.8% (Table 4). In contrast, the baseline model observes a bigger increase in accuracy from 49.1% to 62.6%. While the difference between the two models reduces, the full model remains more accurate. These results also show that the seamless stacking of multiple datasets allowed by our models is effective empirically.

### 5.6 Impact of schema-aware training data

Finally, we investigate to what extent do components introduced by us help in linking when there is training data available that links to the inference KB, \( KB_{test} \). We hypothesize that while attribute-separators will still be useful, attribute-OOV and attribute-shuffle will be less useful as there is a smaller gap between training and test scenarios, reducing the need for regularization.

For these experiments, models from Section 5.4 are further trained with increasing amounts of data from the TAC-KBP 2010 training set. A sample of 200 documents is held out from the training data as a validation set. The models are trained with the exact same configuration as the base models, except with a smaller constant learning rate of \( 2 \times 10^{-6} \) to not overfit on the small amounts of data.

Unsurprisingly, the accuracy of all models increases as the amount of TAC training data in-

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SEP]-separation</td>
<td>62.6 ± 0.8</td>
</tr>
<tr>
<td>attribute-separation</td>
<td></td>
</tr>
<tr>
<td>+attribute-OOV + shuffle</td>
<td>66.8 ± 2.8</td>
</tr>
</tbody>
</table>

Table 4: Adding the Wikia dataset to training improves accuracy of both our model and the baseline, but our model still outperforms the baseline by over 4%.

<table>
<thead>
<tr>
<th>% of TAC training data</th>
<th>[SEP]-sep. w/ reg.</th>
<th>Attribute-sep. w/o reg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>49.1 ± 2.6</td>
<td>61.6 ± 3.6</td>
</tr>
<tr>
<td>1%</td>
<td>62.4 ± 3.1</td>
<td>69.0 ± 0.5</td>
</tr>
<tr>
<td>5%</td>
<td>70.1 ± 2.5</td>
<td>72.8 ± 1.3</td>
</tr>
<tr>
<td>10%</td>
<td>74.5 ± 2.0</td>
<td>76.0 ± 0.8</td>
</tr>
<tr>
<td>25%</td>
<td>80.1 ± 1.2</td>
<td>78.8 ± 0.4</td>
</tr>
<tr>
<td>50%</td>
<td>81.8 ± 1.0</td>
<td>80.5 ± 0.4</td>
</tr>
<tr>
<td>75%</td>
<td>83.1 ± 1.0</td>
<td>81.1 ± 0.2</td>
</tr>
<tr>
<td>100%</td>
<td>84.1 ± 0.6</td>
<td>81.8 ± 0.9</td>
</tr>
</tbody>
</table>

Table 5: Experiments with increasing amounts of training data that links to the inference KB reveal that models with attribute-separators but without any regularization are the most accurate across the spectrum.
Table 6: Categorization of model errors, with an example of each, along with the number of errors observed for each category (out of 100). The underlined text under Mention marks the span containing the actual mention.

<table>
<thead>
<tr>
<th>Category Mention</th>
<th>Gold Entity</th>
<th>Prediction</th>
<th>% Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specific ... Richard Blumenthal of Connecticut and Roy Cooper of North Carolina ...</td>
<td>Connecticut</td>
<td>Hartford, Connecticut</td>
<td>33%</td>
</tr>
<tr>
<td>Generic According to the NY Times, “Foreigners and Lhasa residents who witnessed the violence ...</td>
<td>The New York Times</td>
<td>The New York Times Company</td>
<td>6%</td>
</tr>
<tr>
<td>Context ... the Tombigbee River, some 25 miles (40 kilometers) north of Mobile. It’s on a river ...</td>
<td>Mobile, Ohio</td>
<td>Mobile River</td>
<td>33%</td>
</tr>
<tr>
<td>Related The striker arrived from his native city of Santos in a helicopter ...</td>
<td>Santos (city)</td>
<td>Santos, F.C.</td>
<td>21%</td>
</tr>
<tr>
<td>String the apparent public support for Camara in Guinea had put the organisation in a quandary</td>
<td>Guinea</td>
<td>Senegal</td>
<td>7%</td>
</tr>
</tbody>
</table>

creases (Table 5). As hypothesized, the smaller generalization gap between training and test scenarios makes the model with only attribute separators more accurate than the model with both attribute separators and regularization.

Crucially, the model with only attribute separators is the most accurate model across the spectrum. Moreover, the difference between this model and the baseline model sharply increases as the amount of schema-aware data decreases (e.g. when using 13 annotated documents, i.e. 1% of the training data, we get a 9% boost in accuracy over the model that does not see any schema-aware data). These trends show that our models are not only useful in settings without any data from the target KB, but also in settings where limited data is available.

5.7 Qualitative Analysis

Beyond the quantitative evaluations above, we further qualitatively analyze the predictions of the best model from Table 3 to provide insights into our modeling decisions and suggest avenues for improvements.

5.7.1 Improvements over baseline

First, we categorize all newly correct mentions, i.e. mentions that are correctly linked by the top model but incorrectly linked by the [SEP]-separation baseline by the entity type of the gold entity. This type is one of person (PER), organization (ORG), geo-political entity (GPE), and a catchall unknown category (UKN). This categorization reveals that the newly correct mentions represent about 15% of the total mentions of the ORG, GPE, and UKN categories and as much as 25% of the total mentions of the PER category. This distributed improvement highlights that the relatively higher accuracy of our model is due to a holistic improvement in modeling unseen KBs across all entity types.

Why does PER benefit more than other entity types? To answer this, we count the fraction of mentions of each entity type that have at least one column represented using attribute separators. This counting reveals that approximately 56–58% of mentions of type ORG, GPE, and UKN have at least one such column. On the other hand, this number is 71% for PER mentions. This suggests that the difference is directly attributable to more PER entities having a column that has been modeled using attribute separators, further highlighting the benefits of this modeling decision.

5.7.2 Error Analysis

To identify the shortcomings of our best model, we categorize 100 random mentions that are incorrectly linked by this model into six categories (demonstrated with examples in Table 6), inspired by the taxonomy of Ling et al. (2015).

Under this taxonomy, a common error (33%) is predicting a more specific entity than that indicated by the mention (the city of Hartford, Connecticut, rather than the state). The reverse is also observed

---

10 The 0% results are the same as those in Table 3.

11 This entity typing is present in the KB.
(i.e. the model predicts a more general entity), but far less frequently (6%). Another major error category (33%) is when the model fails to pick up the correct signals from the context and assigns a similarly named entity of a similar type (e.g. the river Mobile, instead of the city Mobile, both of which are locations). 21% of the errors are cases where the model predicts an entity that is related to the gold entity, but is neither more specific, nor more generic, but rather of a different type (Santos Football Club instead of the city of Santos).

Errors in the last category occur when the model predicts an entity whose name has no string overlap with that of the gold entity or the mention. This likely happens when the signals from the context override the signals from the mention itself.

6 Conclusion

The primary contribution of this work is a novel framework for entity linking against unseen target KBs with unknown schemas. To this end, we introduce methods to generalize existing models for zero-shot entity linking to link to unseen KBs. These methods rely on converting arbitrary entities represented using a set of attribute-value pairs into a string representation that can be then consumed by models from prior work.

There is still a significant gap between models used in this work and schema-aware models that are trained on the same KB as the inference KB. One way to close this gap is by using automatic table-to-text generation techniques to convert arbitrary entities into fluent and adequate text (Kukich, 1983; McKeown, 1985; Reiter and Dale, 1997; Wiseman et al., 2017; Chisholm et al., 2017). Another promising direction is to move beyond BERT to other pre-trained representations that are better known to encode entity information (Zhang et al., 2019; Guu et al., 2020; Poerner et al., 2020).

Finally, while the focus of this work is only on English entity linking, challenges associated with this work naturally occur in multilingual settings as well. Just as we cannot expect labeled data for every target KB of interest, we also cannot expect labeled data for different KBs in different languages. In future work, we aim to investigate how we can port the solutions introduced here to multilingual settings as well develop novel solutions for scenarios where the documents and the KB are in languages other than English (Sil et al., 2018; Upadhyay et al., 2018; Botha et al., 2020).

Acknowledgements

The authors would like to thank colleagues from Amazon AI for many helpful discussions that shaped this work, and for reading and providing feedback on earlier drafts of the paper. They also thank all the anonymous reviewers for their helpful feedback.

References


Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of


